A local based approach for path planning of manipulators with a high number of degrees of freedom

Bernard Faverjon, P. Tournassoud

INRIA

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.: (1) 39 63 55 11

Rapports de Recherche

N° 621

A LOCAL BASED APPROACH FOR PATH PLANNING OF MANIPULATORS WITH A HIGH NUMBER OF DEGREES OF FREEDOM

Bernard FAVERJON
Pierre TOURNASSOUD

Février 1987

A Local Based Approach for Path Planning of Manipulators With a High Number of Degrees of Freedom

Bernard Faverjon
Pierre Tournassoud

INRIA
Domaine de Voluceau
BP 105, 78153 LE CHESNAY Cedex
FRANCE

Abstract: This paper presents an alternative to the Potential Field Method of computing local collision-free motions for general manipulators. The main distinction is that we separate the description of the task from constraints of anti-collision. This enables to control accurately all relevant measures of the problem. A task is expressed by the minimization of a function plus eventually some geometric constraints, whereas anti-collision is translated into very simple linear constraints through the methods of the *velocity dampers* and the *tangent separating planes*. This approach is applied to the control of manipulators with a high number of degrees of freedom, using hierarchical descriptions of the environment and the robots. This is illustrated by two realizations, path planning for a ten link manipulator in the cluttered environment of a nuclear plant reactor, and cooperative tasks between two six degree of freedom robots. In the end we discuss how to incorporate the action of a global planner in this general framework.

Une approche locale pour le calcul de trajectoires des manipulateurs à nombre élevé de degrés de liberté

Résumé

Les méthodes globales de calcul de trajectoires pour un manipulateur dans un environnement encombré d'obstacles se heurtent à la complexité exponentielle des algorithmes utilisés.

Dans ce rapport, nous proposons une méthode locale de calcul de trajectoires sûres pour un système robotique, composé d'un ou plusieurs manipulateurs, mettant en jeu un nombre élevé de degrés de liberté. La différence principale avec la classique méthode des potentiels *est que nous séparons la description de la tâche de la prise en compte des contraintes d'anti-collision. Ceci permet de contrôler précisément les diverses mesures critiques du problème. Une tâche s'exprime par la minimisation d'une fonctionnelle et éventuellement par le respect de contraintes géométriques ; la prise en compte des obstacles est traduite en des contraintes linéaires par la méthode des* amortisseurs vitesse *ou des* plans tangents séparateurs.

L'algorithme proposé met en jeu dans une première étape des structures de description hiérarchiques des solides composant l'environnement et les manipulateurs. Ces structures sont mises à profit pour écrire des algorithmes efficaces de tests d'intersection et de calculs de distance entre les objets.

Nous présentons deux réalisations, une pour le calcul des trajectoires d'un manipulateur à dix articulations dans le réacteur d'une centrale, l'autre pour le calcul de trajectoires de coopération entre deux robots.

La méthode étant locale ne permet pas d'affirmer que l'objectif est finalement atteint. Nous proposons de la coupler avec un planificateur global qui raisonne à partir des informations (succès, blocages) résultant de l'exécution de la trajectoire.

1. Introduction

This paper explores the problem of motion planning for robotic systems with a high number of degrees of freedom. It describes basic tools for the calculation of long pieces of trajectories, based on a local view of the environment, and introduces ideas for connecting results of these computations with a global planner.

1.1. Statement of the problem

Global methods for path planning of a manipulator have proved to be successful when the number of degrees of freedom involved is not too high. Known practical algorithms rely on a description of the obstacles in the robot's *Configuration Space* [13,7], chosen among the spaces of independent parameters describing the position of any point bound to a body of the manipulator. Conceptually, planning motions for the robot is then reduced to planning motions for the point representing the configuration.

Such algorithms are composed of two critical steps. The first one is the computation of the Configuration Space obstacles from the available description of obstacles in cartesian space. The latter consists in structuring the *Free Space*, the complement of the obstacles in the Configuration Space, so that the search for a path can be performed efficiently. Lozano-Pérez builds an approximation of a n dimensional Configuration Space by the union of n-1 dimensional *slice projections*, a parameter of the slice representing a range of n dimensional configurations [8,9]. Discretization implies that bodies of the manipulator are augmented of a distance corresponding to legal movements within that range. This method is general: it is used to search a path for the first three links and for reorientation of the hand around a small volume of positions for the wrist. Faverjon builds an octree to represent Free Space for the first three links, while a heuristic approach is used for motions of the hand [2,3]. A cell of the octree is either free, occupied or partially occupied by obstacles, in which case its eight sons give a better description. Turning to profit the hierarchical

structure of the octree provides a very efficient search for a path.

The positive point about these methods is that they are *complete*, meaning there are to provide a path between some origin and goal configurations of the robot, if there exists one, with the restriction that we only deal in practice with conservative approximations of Free Space.

But they have some limits. First, because they are time consuming, they are restricted to off-line path planning and cannot provide on-line control of manipulators. One particular consequence is that it will be difficult to deal with a changing environment. Second, these algorithms are inherently exponential in the number of degrees of freedom of the robot (see [10]), which makes it impossible to deal with some high dimensional special cases. As an illustration of these, we will describe in this paper the following applications:

- path planning for a redundant 10 link 8 degree of freedom robot in the cluttered environment of the reactor of a nuclear plant,
- cooperation of two arms, which implies the control of up to six joints on each robot to be realistic.

A last drawback of the global algorithms is that it is difficult to describe tasks other than simple motion planning within their formalism. Let us take the example of the cooperation between two robots, one arm transfering an object to the other. For this task, we do not care about the precise location where the exchange takes place. Hence, selecting some position, which would be required by a classical planning method, arbitrarily restricts the generality of the task.

On the other hand, local methods are extremely powerful for describing a large family of robotic tasks. Of course their weak point is that the robot can get trapped in a concavity of the obstacles. Another criticism is often formulated. This is these methods rely on the minimization of a function both including a criterion describing the task and terms taking into account the distance to the obstacles, whose effect is to push moving objects away from them. As a single function is made of terms of different nature, it turns out difficult to control independently each one of the relevant measures of the problem.

1.2. Outline of the Approach

We propose here a substitute to these so called *Potential Field* methods (see for example[6]). The main idea is to separate the realization of the task, described by the minimization of a measure of the problem plus eventually some geometric constraints, from the constraints of anti-collision between the robot, the environment, and possibly another moving robot. We build a local model of Free Space as an intersection of free half spaces in Configuration Space, inside which we compute the best move according to the task we have to perform. This makes it possible to respect accurately both the geometric constraints on the task, and anti-collision constraints that remain very close to the geometry of the problem.

The proposed method deeply relies on efficient computation of distances between solids, which is first introduced. Then, after having outlined the approach with the example of the nuclear plant robot, we give a more complete position of the problem and illustrate it with examples of cooperative tasks between two manipulators. In the end, we give ideas for incorporating some features of global planning in this general framework, based on learning.

2. Basic Geometric Tools

2.1. Computing the Distance Between Two Convex Objects

We begin by defining tools for the efficient computation of distances between bodies of the robots and the environment. In the sequel, S_1 and S_2 denote two convex solids in the Euclidian Space, x_1 and x_2 two points belonging respectively to S_1 and S_2, and n a unit vector. Notation $(u \mid v)$ refers to the inner product of vectors u an v.

The Euclidian Distance between S_1 and S_2, equal to $\min_{x_1 \in S_1, x_2 \in S_2} \|x_1 x_2\|$, can be computed by alternatively projecting a point of S_1 onto S_2, the point of S_2 that we obtain onto S_1, etc..., until the distance between the points converges. Yet this method has a major drawback: no thresholding can be done, as the current estimation $\|x_1 x_2\|$ is always bigger than the exact distance, which is unknown.

For this reason, we use the following definition of the distance between two solids.

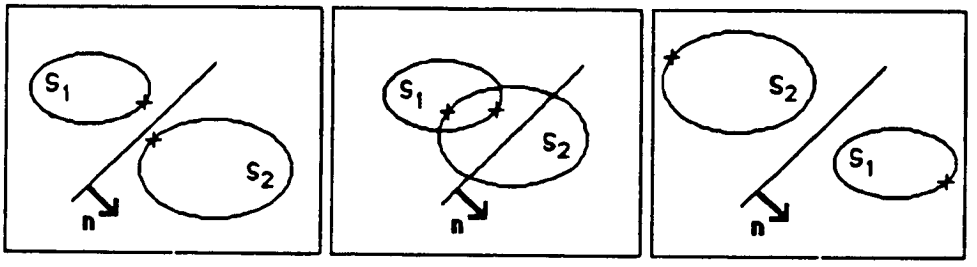
$$d(S_1,S_2) = \max_{\|n\|=1} \min_{x_1 \in S_1, x_2 \in S_2} (n \mid x_1x_2) \qquad (1)$$

With this definition $d(S_1,S_2)$ is equal to the Euclidian Distance only in the case of non overlapping objects. If they do overlap, the "distance" defined by equation (1) becomes negative and measures how far objects interpenetrate.

Definition: *We call distance between* S_1 *and* S_2 *oriented in the direction* n *the scalar*

$$\delta(n) = \min_{x_1 \in S_1, x_2 \in S_2} (n \mid x_1x_2) \qquad (2) \text{ (see Figure 1)}$$

Note that we have $d(S_1,S_2) = \max_{\|n\|=1} \delta(n)$ (3)



$\delta(n)>0$ $\delta(n)<0$ $\delta(n)<0$

Figure 1. Generic examples of the oriented distance.

The interesting point in this definition is that $\delta(n)$ is always lower than $d(S_1,S_2)$. Let us define the influence distance d_i as a threshold on interactions between objects : if objects lie at a distance bigger than d_i, we will not impose any constraint between them. Then if $\delta(n)$ is greater than d_i for some value of n, the same stands for the exact distance and we can declare these objects to be *non interacting*.

Note that the computation of $\delta(n)$ can be decomposed into:

$$\delta(n) = \min_{x_2 \in S_2} (n \mid ox_2) + \min_{x_1 \in S_1} (-n \mid ox_1)$$

for an arbitrary point o. We call x_2 the *first point of* S_2 *in the direction* n. Then x_1 is the first point of S_1 in the opposite direction, -n. To compute the distance $d(S_1, S_2)$ in the case of non overlapping objects, we use the following procedure, illustrated in Figure 2.

1/ *Select an arbitrary point* $\mathbf{x}_1$ *in* S_1.

2/ *Project* $\mathbf{x}_1$ *on* S_2. *Call* $\mathbf{x}_2$ *the projection.*

3/ *Let* $\mathbf{n} = \mathbf{x}_1\mathbf{x}_2 / \|\mathbf{x}_1\mathbf{x}_2\|$. *Compute the first point of* S_1 *in direction* $-\mathbf{n}$, $\mathbf{x}_1'$.

Then $\delta(\mathbf{n}) = (\mathbf{n} \mid \mathbf{x}_1'\mathbf{x}_2)$, and we have

$$\delta(\mathbf{n}) \leq d(S_1,S_2) \leq \|\mathbf{x}_1\mathbf{x}_2\| \qquad (4)$$

The same procedure is run again starting with the point $\mathbf{x}_2$ of S_2 obtained above. It can be shown that $\delta(\mathbf{n})$ and $\|\mathbf{x}_1\mathbf{x}_2\|$ both converge towards $d(S_1,S_2)$ if the procedure is repeatedly applied. In practice, the process is stopped as soon as $\|\mathbf{x}_1\mathbf{x}_2\|-\delta(\mathbf{n})$ is smaller than a given precision, or when $\delta(\mathbf{n})$ is bigger than the influence distance d_i.

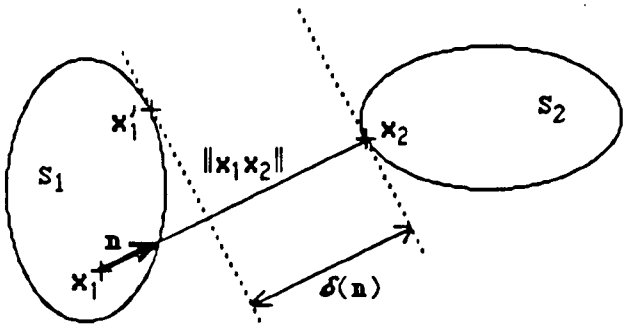


Figure 2. Computing the distance between two solids.
The actual distance lies between $\delta(\mathbf{n})$ *and* $\|\mathbf{x}_1\mathbf{x}_2\|$.

We now consider that S_1 and S_2 are moving solids. Their position and orientation are described by vectors of configuration parameters $\mathbf{q}_1$ and $\mathbf{q}_2$. We note $\dot{\mathbf{u}}$ the time derivative of a vector $\mathbf{u}$. An important property of the distance $d(S_1,S_2)$, when calculated between two moving objects, is that it can always be differentiated.

Proposition: *The time derivative of the distance between moving solids* S_1 *and* S_2 *writes*

$$\dot{d} = (\mathbf{n} \mid \mathbf{J}_2\dot{\mathbf{q}}_2 - \mathbf{J}_1\dot{\mathbf{q}}_1) \qquad (5), \text{ where}$$

- $\mathbf{J}_1$ and $\mathbf{J}_2$ are the jacobian matrices for S_1 and S_2 calculated respectively at points $\mathbf{x}_1$ and $\mathbf{x}_2$ realizing the minimum distance,
- $\mathbf{n}$ is the unit vector on the line $\mathbf{x}_1\mathbf{x}_2$. It realizes the optimum of equation (3).

The proof for this property is given in Appendix I.

2.2. A Hierarchical Description of Solids and Manipulators

In our system we represent solids by unions of simple convex volumes that we call *Primitives*. These are parallelepipeds, prisms, cylinders, truncated cones and spheres. Computation of the distance between two primitives is performed using the procedure given in Section 2.1. The *projection* of a point onto a primitive and the calculation of its *first point in a direction* **n** just require calls to simple functions dependent on the type of the primitive.

To each solid we attach a hierarchical description that we call an *Assembly Tree*. This is a tree whose truncated subtrees give an approximation of the solid by fewer primitives containing the exact description (Figure 3). Please refer to [3] for more precise definitions.

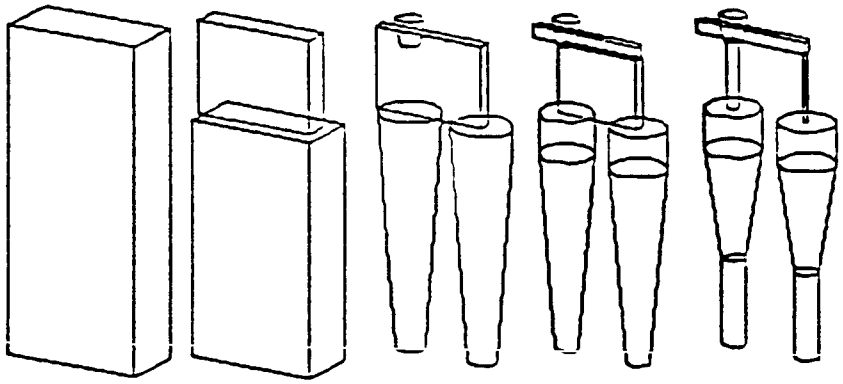


Figure 3. An Assembly Tree representing part of the reactor.
Each subtree is an approximation containing the exact description.

Each body of a manipulator is itself described by an Assembly Tree. To a n degree of freedom manipulator we attach a multi-level description, one level representing the corresponding *Virtual Manipulator* with k degrees of freedom, $0 \leq k \leq n$. A Virtual Manipulator of level k is the manipulator whose last links have been replaced by their swept volumes when joints $k+1$ to n are made free (see Figure 4).

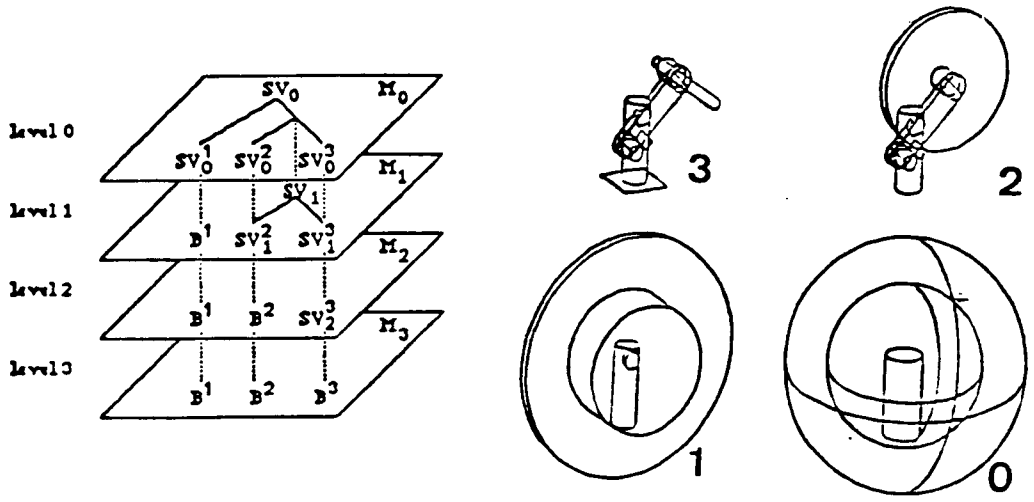


Figure 4. A robot is represented by a stack of n+1 trees.

The tree of level k represents the Virtual Manipulator with k degrees of freedom.

The main advantage of such descriptions, Assembly Trees for solids and Virtual Manipulators for robots, is that they are very efficient for fast intersection tests, and fast computation of distances making use of thresholding. We have described these structures with many details in [4]. In this application, such computations provide at each time increment the list of pairs of primitives that need to be separated: in practice, these are two primitives lying at less than a conservative influence distance function of the maximum cartesian displacement for points on these solids. A pair consists either in one primitive part of the description of a body of a manipulator and one primitive part of a fixed obstacle, or of two primitives on two different robots. From the list of interactions we then compute a local view of obstacles as seen in the Configuration Space of the system.

We will also give two additional applications of these hierarchical structures in Section 5 to underline their generality.

3. Building a Local Model of Obstacles in Configuration Space

3.1. A Substitute to the Potential Field Method

In the so-called *Potential Field Method*, the robot navigates in a potential field sum of a term that attracts it towards the target, and of repulsive terms generated by obstacles. The trajectory is usually obtained by following the gradient of the potential field.

Such methods can work properly in the case there are only few obstacles in the environment. But as they imply that terms of different nature are arbitrarily added in a single function, problems appear in more complex environments. These include:

- oscillations between opposite obstacles,
- arbitrary higher repulsion for two adjacent obstacles than for a unique one,
- impossibility to get close to an obstacle.

In our approach the task is described by a minimization problem plus eventually some geometric constraints. As opposed to the Potentiel Field Method, anti-collision is translated into geometric constraints in Configuration Space and is not included in the function to minimize. This enables to better control all relevant measures of the problem. In the next Section, we describe how to compute anti-collision constraints from the available description of the environment. Let us first show in Figure 5 an example of a trajectory obtained with this algorithm in the complex environment of a nuclear plant reactor, for a redundant 10 link 8 degree of freedom arm carrying a part to be welded. Tasks that can be performed consist in reaching a final position in Configuration Space, or in realizing a goal position and orientation for the grasp frame in cartesian space (Figure 5). Note that in the example illustrated in Figure 5, it was necessary to provide one sub-goal along the trajectory.

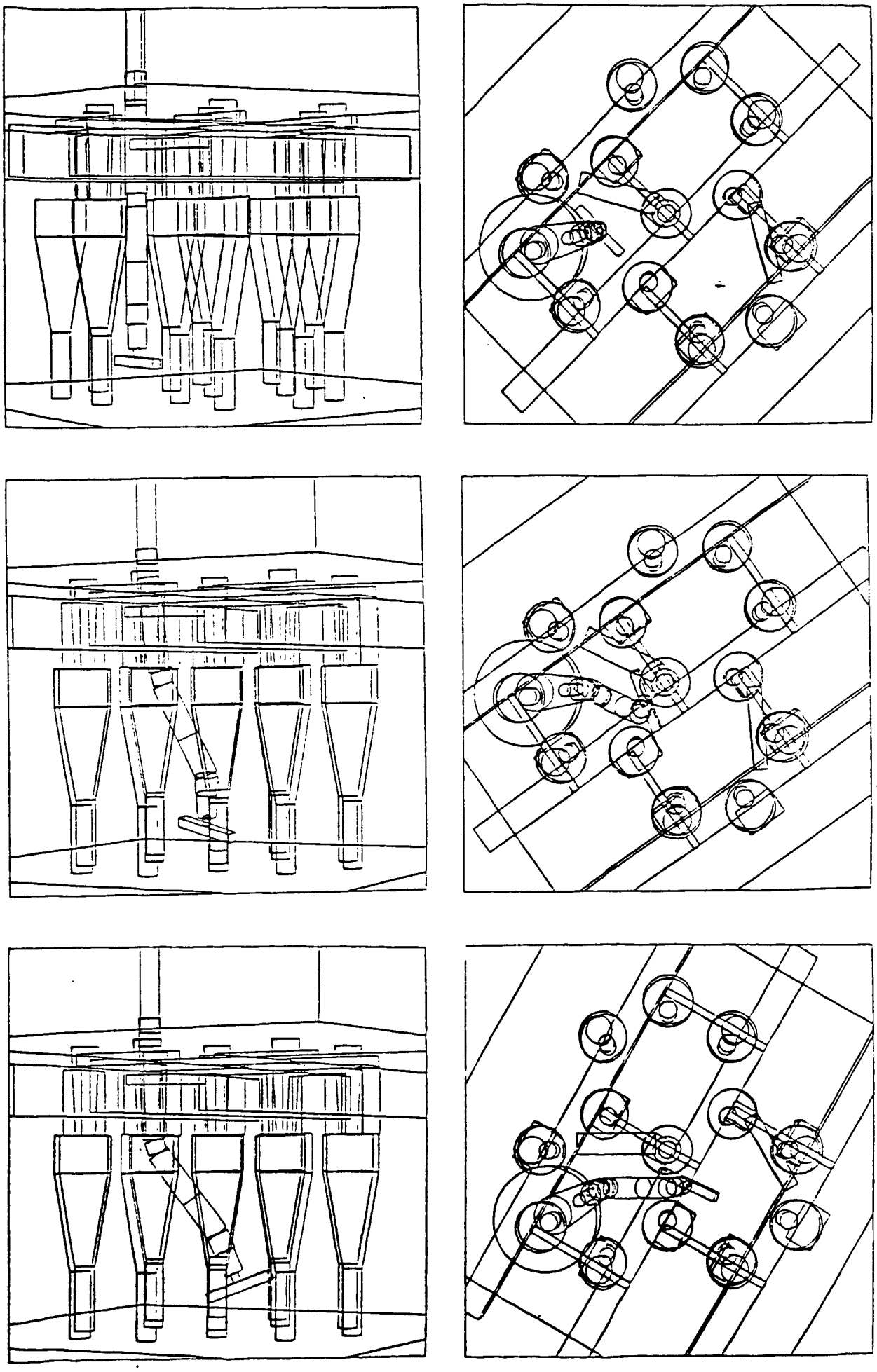


Figure 5 (a). A 10 link robot in the reactor of a nuclear plant.

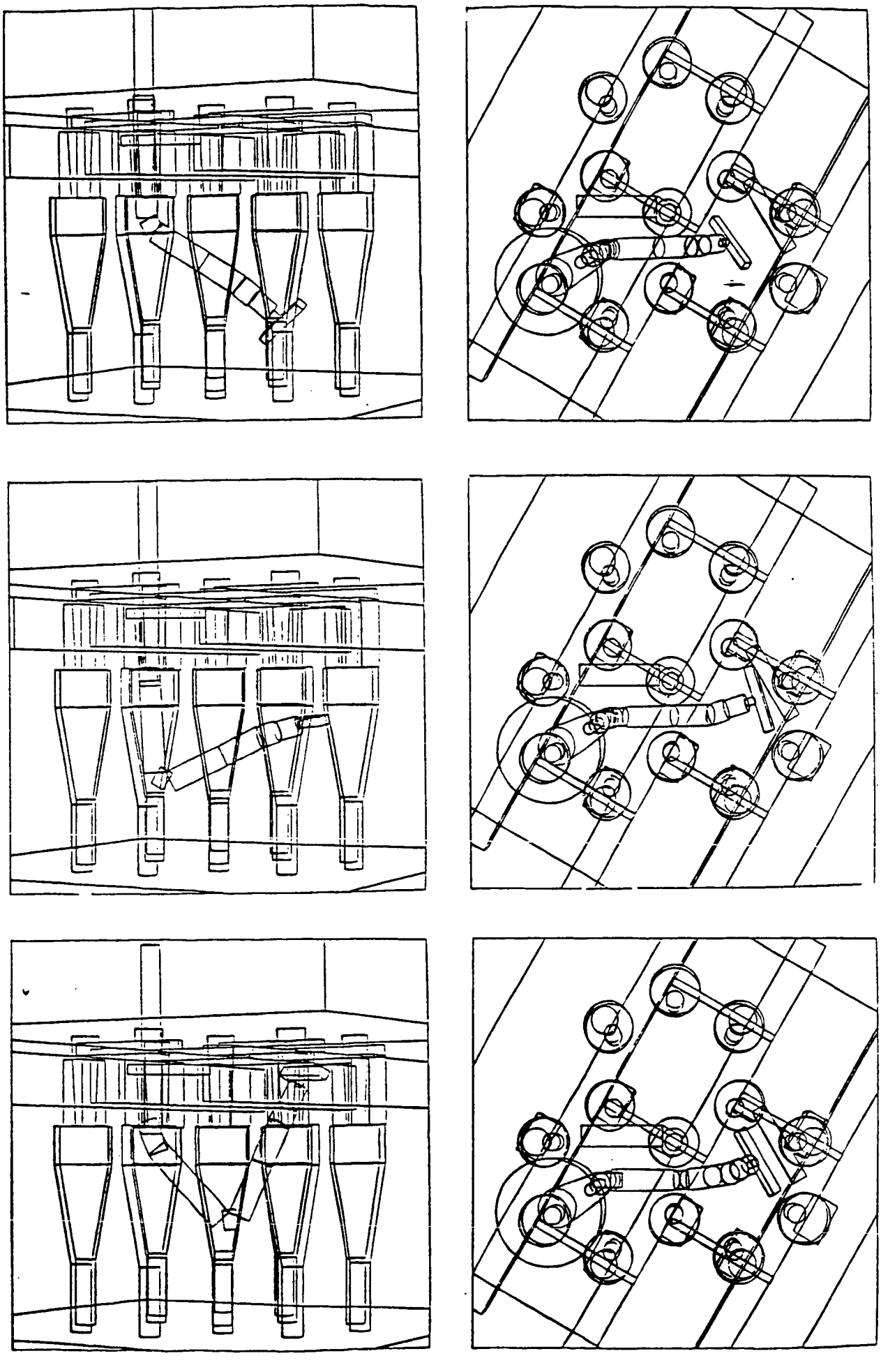


Figure 5 (b).

3.2. The Velocity Damper

In this section we translate an interaction between two convex primitives into a constraint on displacements of the manipulator in Configuration Space. Between a moving primitive and an obstacle we impose a constraint that we call *Velocity Damper* : it expresses that the distance between them must not decrease too fast when it is less than the influence distance d_i . It is shown in Appendix II that there cannot be any collision if we impose:

$$\dot{d} \geq -\xi \frac{d - d_s}{d_i - d_s} \qquad \text{for } d \leq d_i \qquad (6)$$

Here d_s is the security distance at which the robot must stop and ξ a positive coefficient for adjusting convergence speed. A nice property of such constraints is that they allow to go arbitrarily near from the limit $d=d_s$ in a finite time, though the component of the velocity normal to the obstacle gets very small.

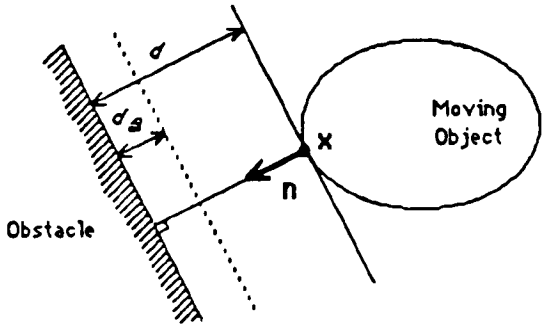


Figure 6. Writing the velocity damper constraint.

We recall (see Section 2.1.) that $\dot{d} = (\mathbf{n} \mid J\,\dot{\mathbf{q}}) = (J^t\,\mathbf{n} \mid \dot{\mathbf{q}})$ where **n** is the unit vector on the line of minimum distance and J the jacobian matrix calculated at point **x** of the moving object realizing the minimum (see Figure 6). Thus inequality (6) can be translated into a simple linear constraint on configuration parameters increments **dq** between time t and $t+dt$, namely with $\mathbf{v}=J^t\mathbf{n}$

$$(\mathbf{v} \mid \mathbf{dq}) \geq -\xi \frac{d - d_s}{d_i - d_s}\, dt \qquad (7)$$

A strong point of this approach is that the generated constraints remain close from the original geometry of the problem. As the minimization we perform will be initiated with

zero joint increments, which clearly respect the constraints, redundant constraints will not be examined in the process and will not have any influence on the result.

3.3. The Case of Two Moving Objects

It is possible to ensure anti-collision between two moving robots using velocity dampers as introduced above. For primitives lying at less than the influence distance we generate the constraint of inequation (6). Linearization yields a constraint similar to (7):

$$(v_1 \mid dq_1) + (v_2 \mid dq_2) \geq -\xi \frac{d - d_s}{d_i - d_s} dt \qquad (8), \text{ with } v_1 = -J_1{}^t n \text{ and } v_2 = J_2{}^t n,$$

J_1 and J_2 being the jacobians of the moving objects calculated at the points realizing the minimum distance. But the simple velocity damper of equation (8) presents no anticipation for anti-collision of two moving objects, as illustrated on Figure 7. For this reason, we introduced the *Tangent Separating Plane* method for computing anti-collision trajectories of two moving objects. It is outlined in [12,4] and described with more details in [11]. In short, it consists in forcing the bodies of both manipulators to slide on a chain of possibly moving virtual obstacles that separate them. For a pair of interacting primitives this obstacle is chosen among the planes that separate them so that it least perturbs their nominal displacements. This in particular implies it will be tangent to both objects.

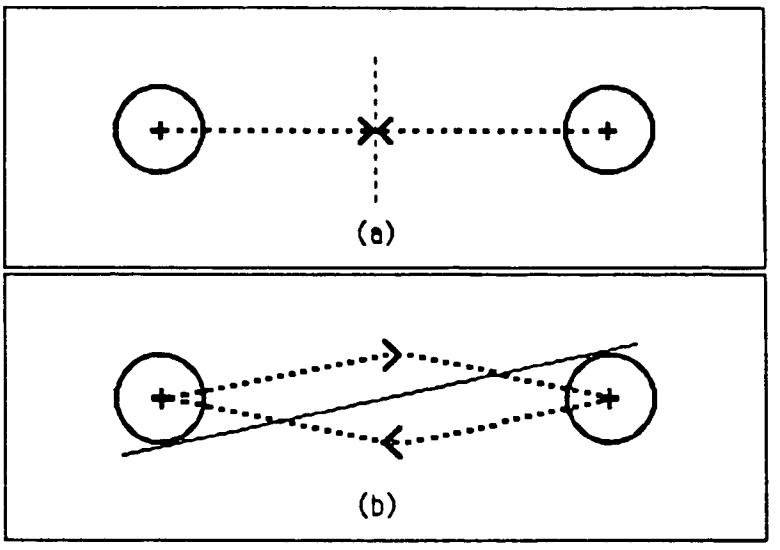


Figure 7. Exchange of the positions of two discs in the plane.
Dotted lines represent trajectories of the discs.
- *The velocity damper method (a) causes a dead-lock at mid-distance.*
- *Sliding on the tangent separating line (b) produces optimal trajectories for both robots.*

The important point here is that the constraints generated by using this method again can be translated into simple linear constraints on configuration parameters increments that write :

$$(v_1 \mid dq_1) + (v_2 \mid dq_2) \geq 0 \qquad (9)$$

This method provides safe trajectories for two arms moving independently, with few cases of dead-locks.

4. General Position of the Problem

The realization of the task is expressed by the minimization of a measure of the problem, plus eventually the respect of some geometric constraints. Anti-collision constraints are expressed separately by means of the velocity damper or the tangent separating plane constraint (in the case of two moving robots) introduced above.

4.1. Description of a Task

Let θ^i, $i \in \{1,...,n\}$, be configuration parameters of the problem. We use homogeneous configuration parameters $\mathbf{q}=(q^1,...,q^n)$ such that $q^i = \theta^i / \omega^i$ for $i \in \{1,...,n\}$, where ω^i is the maximum value of the i-th configuration parameter derivative. Hence, with the norm $\|\mathbf{q}\| = \max_{i=1...n}|q^i|$ for the vector of configuration parameters, we always have $\|\dot{\mathbf{q}}\| \leq 1$.

A task is described by the control of a measure of the problem, a vector $\tau(\mathbf{q})$ of a p-dimensional space. We again write its norm $\|\tau(\mathbf{q})\|$ for simplicity. We suppose that we want to reach a state $\mathbf{q}$ such that $\tau(\mathbf{q})=0$. This is always possible by adding a constant vector to τ. Such a state will be called a solution of the problem. For example, if the task simply consists in reaching a goal configuration $\mathbf{q}_g$, we write $\tau(\mathbf{q})=\mathbf{q}-\mathbf{q}_g$.

The first step consists in translating this task into an optimization problem. We observe that if we simply minimize $\|\tau(\mathbf{q})\|$, we move too early on the degrees of freedom that are not constrained, which augments eventualities of dead-locks.

Let $\delta\tau_{max}$ be the maximum value for the measure derivative. In practice, we make use of a conservative approximation of $\delta\tau_{max}$. Writing that we try to follow a line in the space where the task is expressed, we formulate the problem:

$$(*) \text{ minimize } \frac{1}{2} \|\dot{\tau}(\mathbf{q})-\dot{\mathbf{I}}\|^2, \text{ with}$$

$$\dot{\mathbf{I}} = - \delta\tau_{max} \tau(\mathbf{q}) / \|\tau(\mathbf{q})\|$$

Note that $\dot{\mathbf{I}}$ is not the derivative of the measure along a reference trajectory, but only the desired value of $\dot{\tau}(\mathbf{q})$. In particular we do not memorize eventual deviations caused by an obstacle.

In the case of a goal in Configuration Space, this translates easily in terms of desired configuration parameters increments :

$$\mathbf{dq} = \frac{\mathbf{q}_g - \mathbf{q}}{\max_{i=1...n} |q_g^i - q^i|} dt$$

In the absence of any obstacle the trajectory thus follows a line in Configuration Space.

The second step consists in linearizing the variations of the controlled measure around current joint angles. With $\mathbf{dq}=(dq^1,...,dq^n)$ the vector of configuration parameters increments we write:

$$\dot{\tau}(\mathbf{q}) = \Sigma_{i=1...n} (\partial\tau/\partial q^i)\, dq^i/dt.$$

Minimization (*) is rewritten:

(**) minimize $1/2\ (\mathbf{dq} \mid \mathbf{A}\ \mathbf{dq}) + (\mathbf{b} \mid \mathbf{dq})$, where

- **A** is the symmetric n×n matrix of general term $a_{ij} = (\partial\tau/\partial q^i \mid \partial\tau/\partial q^j)$,
- **b** is the n dimensional vector of general term $b_i = (-\dot{\mathfrak{T}}\, dt \mid \partial\tau/\partial q^i)$.

We add the constraints of anti-collision (velocity dampers or tangent separarating planes):

$(\mathbf{v}_c \mid \mathbf{dq}) \geq \alpha_c\, dt$, for $c \in \{1,...,n_c\}$, n_c being the total number of generated constraints, and bounds on variations of the joint angles:

$\|\mathbf{dq}\| \leq dt$ this is $-dt \leq dq^i \leq dt$ for $i \in \{1,...,n\}$.

We thus have reduced the control of the measure $\tau(\mathbf{q})$ to a simple minimization problem with *quadratic criterion* and *linear constraints.*

In the case we simultaneously want to realize k sub-tasks $\tau_1(\mathbf{q}),...,\tau_k(\mathbf{q})$, the composed task is simply the vector $\tau(\mathbf{q})=(\tau_1(\mathbf{q}),...,\tau_k(\mathbf{q}))$ written in the space cross-product of the spaces of the sub-tasks. We can easily *prioritize* tasks by adding weighting coefficients p_i and write $\tau(\mathbf{q})=(p_1\tau_1(\mathbf{q}),..., p_k\tau_k(\mathbf{q}))$.

4.2. The Example of Cooperative Tasks Between Two Manipulators

A straightforward application of this method is the problem of reaching a goal configuration, or realizing a transform for the grasp frame in cartesian space.

We now give some applications for two manipulators sharing common workspace, with respectively n_1 and n_2 degrees of freedom. We write $\mathbf{q}_1$ and $\mathbf{q}_2$ their configuration vectors, and $\mathbf{q}=(\mathbf{q}_1,\mathbf{q}_2)$ the n_1+n_2 dimensional configuration vector of the system.

We have described avoidance of two arms with many details in [11]. For related works on this problem see [1,5]. In this case the measure we command simply writes $\tau(\mathbf{q})=\mathbf{q}-\mathbf{q}_g=(\mathbf{q}_1-\mathbf{q}_{1g}, \mathbf{q}_2-\mathbf{q}_{2g})$. Notice that in such problems where the task can be uncoupled into $(\tau_1(\mathbf{q}_1), \tau_2(\mathbf{q}_2))$, the priorities we impose on the task by writing

$\tau(q)=(p_1\tau_1(q_1), p_2\tau_2(q_2))$ have the intuitive effect of favoring one robot respective to the other. Indeed in the case of independent motions of two arms, the arm with lower priority will have to make more efforts to avoid collision.

We now concentrate on tasks of cooperation between two arms. The first task we will describe is the transfer of an object from one arm, say $Robot_1$, to the opposite one, $Robot_2$. Transfer will be performed if we realize a relative position and orientation between their grasp frames. It can be specified using the following sub-tasks τ_t and τ_θ.

- τ_t : Make the position of a point bound to the grasp frame of $Robot_1$ and that of a point bound to the grasp frame of $Robot_2$ coincide. Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be the respective positions of these points in an absolute frame. The measure we control is $\tau(q)=\mathbf{x}_2(q_2)-\mathbf{x}_1(q_1)$.
- τ_θ : Align two axes, one bound to each grasp frames. Let $\mathbf{u}_1$ and $\mathbf{u}_2$ be the unit vectors of these axes in an absolute frame. We command the measure $\tau(q)=\mathbf{u}_2(q_2)-\mathbf{u}_1(q_1)$.

Calculations of the corresponding linearized criteria are given in Appendix III.

A relation of type τ_t induces 3 scalar constraints. If we add a *compatible* relation of type τ_θ, we impose 5 constraints. Here compatible means there exists an isometry between points and axes of $Robot_1$'s frame and those of $Robot_2$. By again aligning two axes which define compatible relations with the ones above, we impose 6 constraints in all. Hence for a given grasp of $Robot_1$ on the object, the position and orientation ot the grasp frame of $Robot_2$ are specified relative to a frame bound to the object. Note that nevertheless, in the case of two six degree of freedom manipulators, regrasping of an object does not imply a specific location for the object in an absolute frame.

Once these relations are realized, we can maintain them as geometric constraints included in the definition of the task. This describes such a task as two cooperating robots carrying a heavy load. We impose the constraint $\tau(q)=0$, that we linearize around current value $\tau(t)$:

$$\tau(t)+ \Sigma_{i=1\ldots n} (\partial\tau/\partial q^i)\, dq^i = 0.$$

For example, fixing the position of the grasp point of one robot relative to the grasp frame of the opposite one, and aligning two pairs of axes, is equivalent to specifying the grasps of both robots on one object. For two six degree of freedom robots we still have six

degrees of freedom left. Hence it is possible to specify a final configuration to one of the robots which will be called the *leader*. The opposite arm is called the *follower*, as its motion would be completely determined once the motion of the leader and the constraints on the task are defined. In fact our controller will compute the best motion for both arms respecting the constraints on the task and avoiding obstacles. Another possible task is to control directly the final position and orientation of the object, which is equivalent to giving a goal position and orientation for the grasp frame of one of the robots, respective to an absolute frame.

5. Results

This new approach has been successfully tested using hierarchical CAD models of the robots and the environment. Figures 9 and 10 illustrate applications to the avoidance of two six degree of freedom arms performing independent motions, and to cooperative tasks between these robots. Computation time (on a SUN 3 workstation) is still about 10 times slower than what it should be for real-time avoidance and cooperation of these arms.

The use of the hierarchical description introduced in Section 2.2. constitutes a very powerful tool towards real-time application of our method. We give two examples.

The first example concerns the use of Virtual Manipulators in place of the actual robot for some applications. For instance, avoidance of two arms can be performed using Virtual Manipulators with only 3, 4, or 5 degrees of freedom, using swept volumes when the last joints are free instead of the exact description of the hand (see Figure 8). We thus generate less interactions between primitives along a trajectory, which yields great savings of computation time.

Hierarchical descriptions are also used for efficiently updating the current model of the environment when the robot is moving. This model consists in a list of nodes from the Assembly Tree describing the environment. When a primitive of the robot moves towards a given node, a more precise description is obtained by replacing this node by its sons in the tree. On the contrary, the robot may move away from an obstacle so that two nodes with

the same father now lie at a distance greater than the threshold d_i from the robot. Then they are replaced by their father in the list. In the application illustrated in Figure 5, the environment is composed of about 100 primitives and the robot of 10. A straightforward use of the hierarchical structures enables to deal with only 50 interactions at a time, compared with the 1000 potential ones. Furthermore, most of the corresponding primitives of the environment lie at a distance superior to the threshold d_i from the primitive on the robot. We can perform a worst case updating of the distance using a conservative approximation of the maximum velocity of points on the moving primitive. Then only the nodes lying at an estimated distance lower than d_i from the robot need to be examined precisely at next step, typically 10 at each time increment. This enables real-time computation of collision-free trajectories for this particular manipulator, which moves only at a maximum velocity of 20 *cm* per second at the tip of the hand.

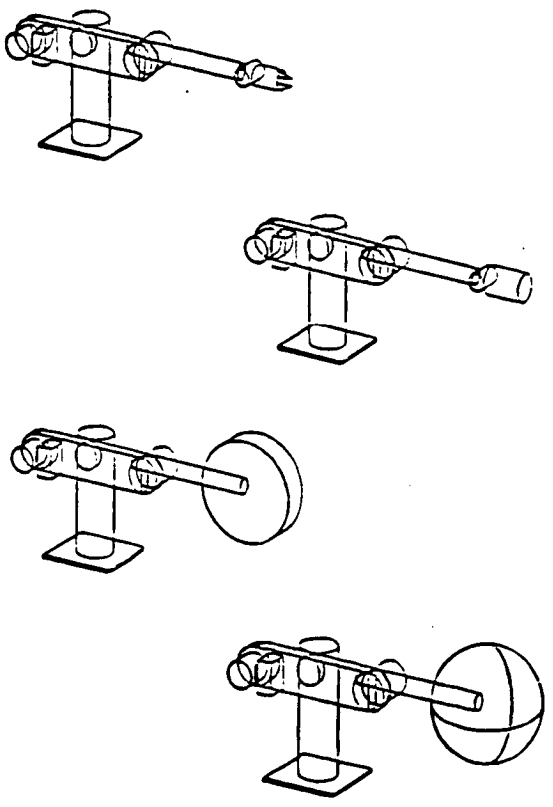


Figure 8. Computing trajectories for virtual manipulators with 3, 4 or 5 degrees of freedom reduces the number of interactions with the environment.

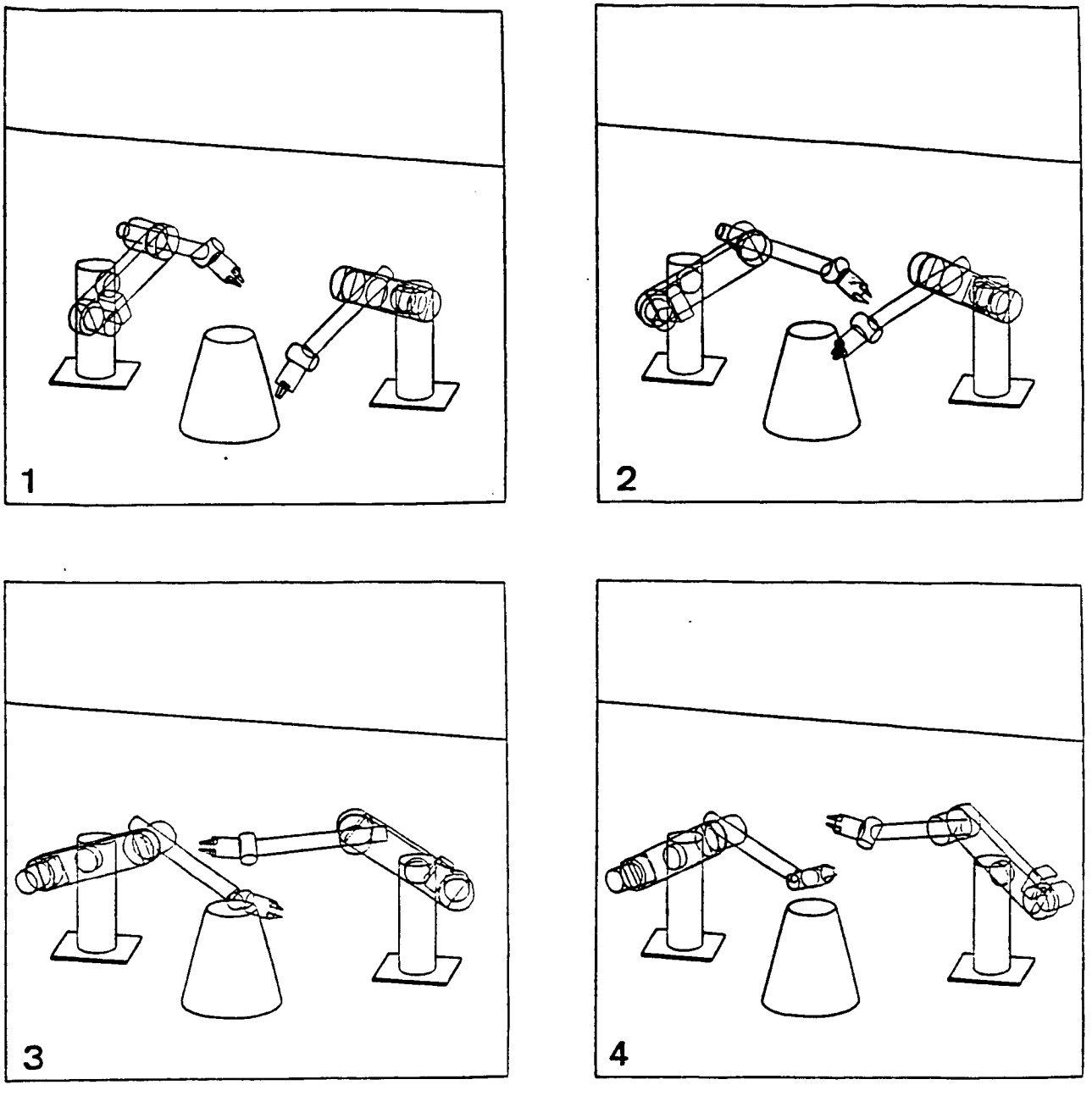
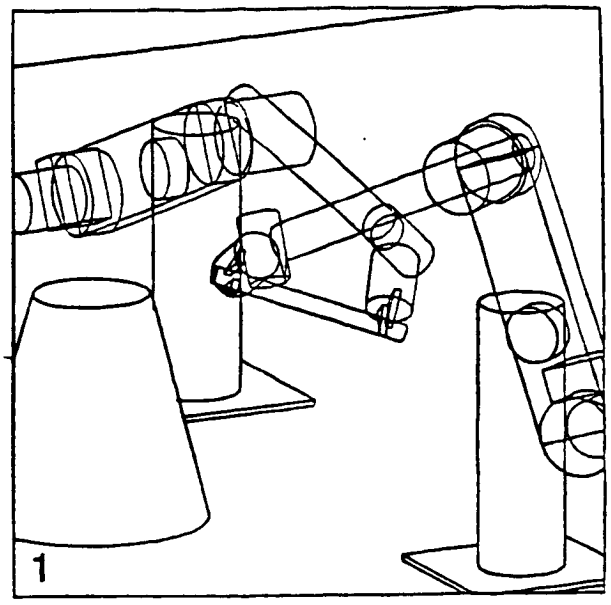


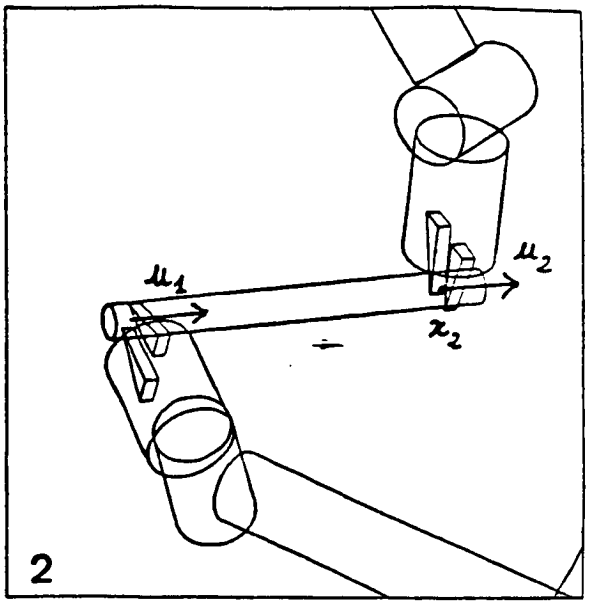
Figure 9. Avoidance of two arms using the Tangent Separating Plane method.

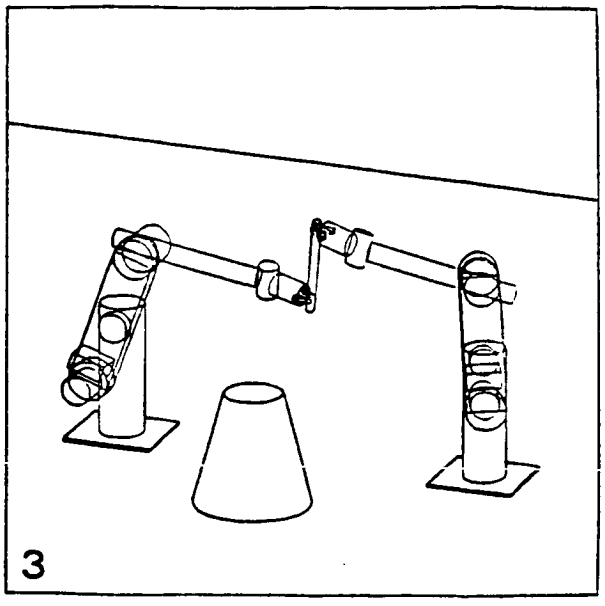
1: Initial positions.

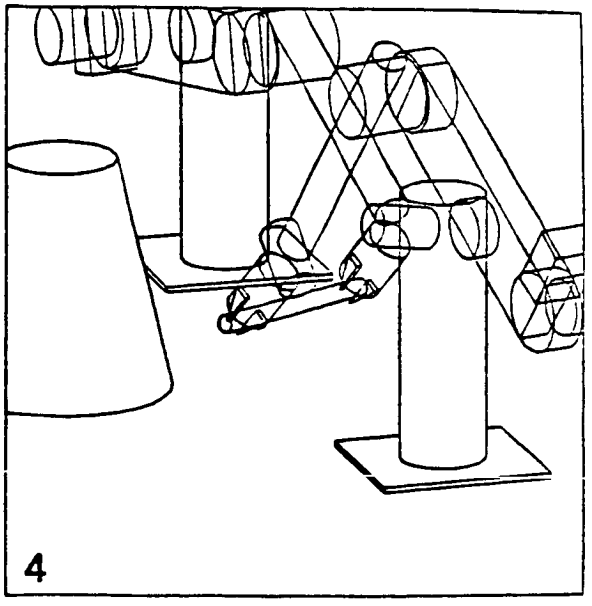
5: Final positions.

We command virtual robots with 5 degrees of freedom each. 20 interactions between primitives of the robots or the environment are generated at most.









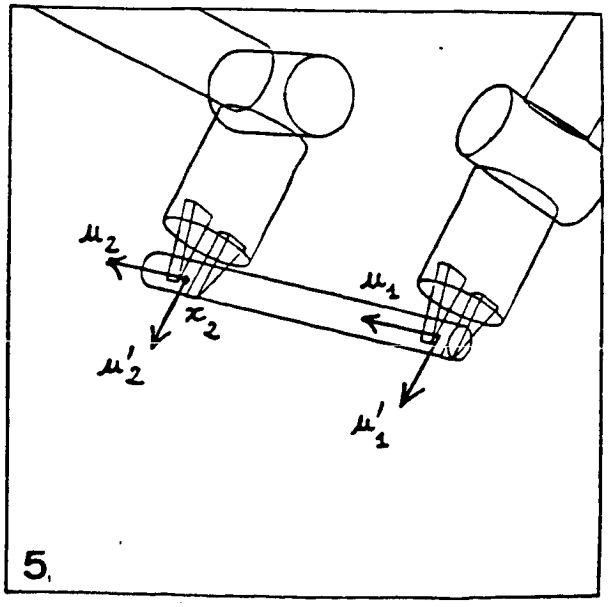


Figure 10. Cooperation of two manipulators

- *1,2. Transfer of a rod. Only one relation τ_t and one relation τ_θ need to be specified because of the symmetry of the task.*
- *3,4,5. Two robots carrying one rod. The transform between the grasp frames needs to be totally specified.*

6. From Local Methods to Global Planning: Learning

Global methods give rise to memory space and computational time exponential in the number of degrees of freedom. Thus they prove to be impractical for systems with a high-dimensional Configuration Space. Indeed, in order to construct a representation of Free Space, ranges of configuration parameters are recursively cut into fine slices. On the contrary, this local method yields long pieces of safe trajectories even when little free space is available. Nevertheless, local path computation does not guarantee that the trajectory we obtain constitutes the task minimum. More, it eventually requires that some more global information such as sub-goals be given to avoid dead-ends generated by concave arrangements of obstacles.

The new approach we propose is a combination of local and global methods, based on learning. It makes use of a graph structure to describe global knowledge about the environment, but no explicit geometric information is stored at this level.

Nodes represent relatively large cells of the Configuration Space of the robot(s), but no classification is done according to whether they are free or not. Each node has a weight associated to it, defined as the probability for a local method to succeed in making the robot enter this cell when coming from one of its neighbors. In the absence of any knowledge, weigths are initialized with equal a priori probabilities.

Suppose we want to compute a trajectory between an initial and a final configuration. We first perform a best-first-search in the graph from the node containing the initial position to the node containing the goal. We use a A* algorithm maximizing the product of the probabilities of the cells along the path. The heuristic cost of a node we open is an estimate of the norm $\|\tau(q)\|$ of the measure we want to minimize for configurations inside this cell, weighted by the average probability for all cells of the graph. This yields a global path or sequence of cells with the highest probability of success.

We then run some local algorithm to move the robot, taking as sub-goal the center of the next cell on the global path as soon as we enter one cell. Updating of the probabilities along the path is done at this time. In the case of a failure of the local method at one point

(this is a dead-lock), a new search is performed from that point in the updated graph.

This will eventually produce a satisfactory trajectory, or we will declare there is no path between these configurations. On the way, we have memorized some knowledge on this part of the environment.

7. Conclusion

The local algorithm described above has been tested for a number of robotic cells using hierarchical CAD models of the environment and the manipulators. The particular application of the nuclear plant robot has been verified on site. Trajectories produced off-line with our algorithm have been successfully reproduced, the robot moving as close as $1cm$ from the obstacles in some very constrained configurations. The on-line control of robots in cells similar to those described in Figure 9,10 seems a reasonable goal.

The connection of this algorithm with a global planner as described in Section 6 is under implementation. Details will be reported elsewhere.

Note that such an approach will not however exclude the need for standard global methods of planning in very constrained parts of the environment. Indeed the graph in which we store information from learning must be a *loose* graph, each node having up to $2n$ neighbors if n is the number of commanded degrees of freedom of the system. The global approach can also be used in the context of learning for providing a first updating of the probabilities we attach to the nodes of the graph. It will be performed using virtual manipulators so that the number of degrees of freedom is within the reach of the global algorithm.

A promising idea is to make local and global approaches work together by labelling regions of the Free Space of the robots in term of what method should be used for computing the trajectory. The local method described above would be dedicated to regions where the environment is not too complex and where the robots can eventually cooperate.

Labyrinth-like parts of the environment would require off-line planning using a mixture of global methods and of the learning approach described above.

Appendix I

We write $d = (\mathbf{n} \mid \mathbf{x}_1\mathbf{x}_2)$ for unit vector $\mathbf{n}$ and points $\mathbf{x}_1$, $\mathbf{x}_2$ verifying the minimum distance. Thus vector $\mathbf{n}$ is directed along the line joining $\mathbf{x}_1$ and $\mathbf{x}_2$ and points from S_1 towards S_2. Note that the point $\mathbf{x}_i$ of S_i verifying the minimum distance is not a point bound to the solid, but describes a trajectory on its surface. Hence its time derivative is equal to:

$\dot{\mathbf{x}}_i = \mathbf{v}_i + \dot{\mathbf{x}}_i/S_i$, with

- $\dot{\mathbf{x}}_i/S_i$ the relative velocity of this point in a frame bound to S_i,
- $\mathbf{v}_i$ the velocity of the point bound to S_i coinciding with $\mathbf{x}_i$ at time t.

The time derivative of the distance writes $\dot{d} = (\mathbf{n} \mid \dot{\mathbf{x}}_2 - \dot{\mathbf{x}}_1) + (\dot{\mathbf{n}} \mid \mathbf{x}_1\mathbf{x}_2)$. The second inner product is equal to zero as the norm of $\mathbf{n}$ remains constant. The first inner product is also equal to $(\mathbf{n} \mid \mathbf{v}_2 - \mathbf{v}_1)$. Indeed as point $\mathbf{x}_i$, i=1,2, describes a trajectory on the surface of S_i the inner product $(\mathbf{n} \mid \dot{\mathbf{x}}_i/S_i)$ is constantly equal to zero. We derive equation (5)

$\dot{d} = (\mathbf{n} \mid J_2\dot{\mathbf{q}}_2 - J_1\dot{\mathbf{q}}_1)$

with J_1 and J_2 the jacobians for solids S_1 and S_2 at points $\mathbf{x}_1$ and $\mathbf{x}_2$. For a single moving object, we simplify this expression into: $\dot{d} = (\mathbf{n} \mid J\,\dot{\mathbf{q}})$.

Appendix II

First we prove that the constraint generated by inequation (6) $\dot{d} \geq -\xi \dfrac{d - d_s}{d_i - d_s}$, with $\xi \geq 0$, ensures there will be no collision between the moving objects.

Let us write $F(t)$ the value at time t of the expression $(d - d_s)\, e^{\xi t / (d_i - d_s)}$

Inequation (6) implies $\dot{F}(t) \geq 0$.

We derive we have $F(t) \geq F(0)$ for any time t, or equivalently $d \geq d_s + F(0)\, e^{-\xi t / (d_i - d_s)}$

Two objects are said to be separated if they verify condition $d \geq d_s$. As the objects are separated at time 0 we derive $F(0)=d_{t=0}-d_s$ is positive, and so they remain separated later.

Note that it is possible to go arbitrarily near from the constraint $d=d_s$ in finite time, though the component of the velocity normal to the constraint gets very small. Variations of factor ξ enable to adapt convergence speed.

Appendix III

• A relation of type τ_t is defined by $\tau(q)=x_2(q_2)-x_1(q_1)$. Let J_1 be the jacobian matrix for $Robot_1$ at point x_1, J_2 for $Robot_2$ at point x_2. We linearize variations of x_1 and x_2:

$$x_1 = x_{1t} + J_1\, dq_1 \text{ and } x_2 = x_{2t} + J_2\, dq_2.$$

We write $J_r = (-J_1\ J_2)$ the *relative jacobian* for $Robot_1$ at point x_1 and $Robot_2$ at point x_2. Matrix A and vector b of optimization problem (**) (see Section 4.1.) then write :

$$A = J_r^t J_r \qquad \text{and} \quad b = (x_{2t} - x_{1t} - \dot{\tau}\, dt)^t J_r$$

• A relation of type τ_θ yields identical equations with $\tau(q)=u_2(q_2)-u_1(q_1)$.

References

[1] **Erdmann, M. and Lozano-Pérez, T.** 1986. On Multiple Moving Objects. A.I. Memo No. 883, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

[2] **Faverjon, B.** 1984. Obstacle Avoidance Using an Octree in the Configuration Space of a Manipulator. Proceedings of the IEEE International Conference on Robotics and Automation, Atlanta, pp. 504-512.

[3] **Faverjon, B.** 1986. Object Level Programming of Industrial Robots. Proceedings of the IEEE International Conference on Robotics and Automation, San Fransisco, pp. 1406-1411.

[4] **Faverjon, B. and Tournassoud, P.** 1986. A CAD System for Multi-Robot Coordination. Proceedings of the NATO International Advanced Research Workshop on Languages for Sensor-based Control in Robotics, Pisa.

[5] **Freund, E. and Hoyer, H.** 1984. Collision Avoidance for Industrial Robots with Arbitrary Motion. Journal of Robotic Systems, Vol. 1, No.4.

[6] **Kathib, O. and Le Maitre, J. F.** 1978. Dynamic Control of Manipulators Operating in a Complex Environment. Proceedings of the 3rd CISM-IFToMM, Udine.

[7] **Lozano-Pérez, T. and Wesley, M. A.** 1979. An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles, Comm. of the ACM, Vol. 22, No. 10, pp. 560-570.

[8] **Lozano-Pérez, T.** 1983. Spatial Planning: a Configuration Space Approach. IEEE Transactions on Computers, Vol. C-32, No. 2, pp. 108-120.

[9] **Lozano-Pérez, T.** 1986. A Simple Motion Planning Algorithm for General Robot Manipulators. A.I. Memo, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

[10] **Schwartz, J.T. and Sharir, M.** 1982. On the Piano Mover's Problem II. General Techniques for Computing Topological Properties of Algebraic Manifolds. Technical Report No. 41, New York University Computer Science Department, Courant Institute of Mathematical Sciences.

[11] **Tournassoud, P.** 1986. On Motion Coordination. INRIA Research Report No. 549.

[12] **Tournassoud, P.** 1986. A Strategy for Obstacle Avoidance and its Application to Multi-Robot Systems. Proceedings of the 1986 IEEE International Conference on Robotics and Automation, San Fransisco, pp. 1224-1229.

[13] **Udupa, S.** 1977. Collision Detection and Avoidance in Computer Controlled Manipulators. Proceedings of the 5th International Joint Conference on Artificial Intelligence.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique